**2.Variables**

* 1. IntroductionPage 1
  2. Tokens and their typesPage 3

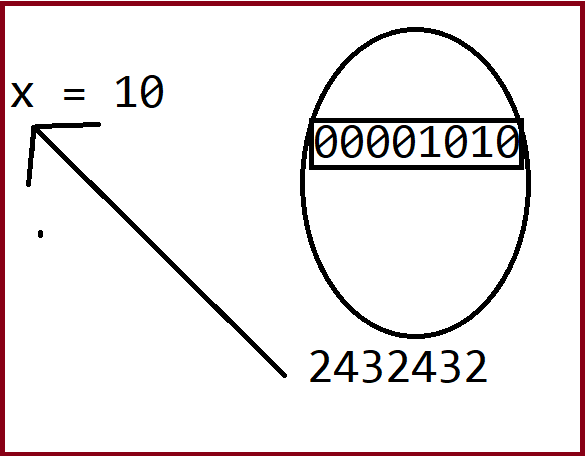
x= 10

For ex :Binary values from decimal number

x = 10 : Writing data 00001010 <-> 1 byte

00001010

|  |
| --- |
| 0 0 0 0 1 010 |



Print(x) : Read operation

**x = 10 : Write operation**

=========================

Step1 : First python starts executing RHS

Step2 : It will check whether value exists or any expression

=>**If value**

-> Here value 10 will be converted to binary format 1010

-----------------

|0|0|0|0|1|0|1|0|

-----------------

**=>If expression**

-> First it simplifies the expression,gets final value

-> Then follows above procedure

Step3: Binary format address(memory allocation address) will be given to variable

**print(x) : Read operation**

==========================

- Python will go to the address of x variable

- It will take the binary value from that address and converts to decimal format

- Returns value to console

**Data types:**

1. **Numbers:**

**integer :**10, 123, 432

**float:**32.43, 543.56, 3456.34

**long:**7406900500, 324324324324(Removed)

**complex:**4+5j(Will never use)

**binary 2octa 8 deca 10 hex 16**

1. **Boolean:**

**True - 1 bit 1**

**False - 1 bit 0**

**x = 10**

**LHS = RHS**

**X = 10+20+30**

**X = 10 1010**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** |

**7 6 5 4 3 2 1 0**

**576543424**

**x = 10**

**LHS = RHS**

**- We are assigning the value 10 to variable 'x'**

**value = 10**

**variable = x**

**- Program execution starts from right to left**

**- LHS ->should always be a variable**

**- RHS ->finally should become value**

**(end result/final output)**

**- Digital 0 1 binary format(number,audio,video,image,content)**

**1 byte - 8 bits0 00000000 to 255 11111111**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** |

**1024 bytes - 1KB**

**1024 KB - 1MB**

**1024 MB - 1GB**

**| | | | | | | | |**

* Variable is a name which is used to refer **memory location of value**.
* Variable also known as **identifier** and used to hold value.
* A variable, as the name indicates is something whose value is changeable over time.x = 10
* In Python, **we don't need to specify the type of variable(int,float)** because Python is a type infer language and smart enough to get variable type.

x = 10 int x = 10;

y = 11.2

10 + 15.5 => 10.0 + 15.5 => 25.5

* **Rule :**Variable names can be a group of both letters and digits, but they have to begin with a letter or an underscore.

**\_abcabcx \_x123 \_123**

**123 🡪 Wrong**

* It is recommended to use **lowercase letters** for variable name. Rahul and rahul both are two different variables.

#### Note - Variable name should not be a keyword.

* Based on the **data type of a variable**, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

**x = 10 # Python**

float x = 10.5 # Java

int x = 10

boolis\_active = True

2L 5L MemoryWater

---------------------------------------------

int x = 10=>2L can 🡸2Lwater

float x = 10.5 => 5L can 🡸 5L water

**float x = 10** =>5L can🡸2L water

int x =10.5=> 2L can 🡸5L water

Implicit casting(OK), Explicit casting(XXX)

1.Python interpreter allocates memory(2 bytes)

2. Converts to binary format (00010100)

3. Copy above binary data to reserved memory

4. Give address of above memory location to variable **“x”**

## Declaring Variable and Assigning Values

## JAVA/Others :

## int x; Declaration of variable

## x = 10 Initialization

## =========================================

## int x = 10; Initialization

## =========================================

## expression vs equation

## x2+y+z --- expression

## x2+y+z = 100 --- equation / statement in python

## Python :

## x = 10 Initialization of variable x

Python does no**t bound us to declare variable** before using in the application. Python variables **do not need explicit declaration** to reserve memory space.Itallows us to create variable at required time.

We don't need to declare explicitly variable in Python. When we assign any value to the variable that variable is declared automatically.

The equal **(=)** operator is used to assign **value** to a **variable**.

## The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

## For Ex :age = 20 x = 20

## In above ex. operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable. For example

e\_count=10# An integer assignment

miles=2.5# A floating point

emp\_name="Madhu"# A string

print(e\_count)

ord\_refno= 12214324324324

System.out.println(counter); # JAVA

printf(miles)

print(miles)

print(name)

Here, 100, 1000.0 and "Madhu" are the values assigned to counter, miles, and name variables, respectively. This produces the following result −

100

1000.0

Madhu

## Multiple Assignment

## i`. Assigning single value to multiple variables

Python allows you to assign a single value to several variables simultaneously. For example −

a = b = c = 10

Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location.

**ii.Assigning multiple values to multiple variables:**

You can also assign multiple objects to multiple variables. For example −

a,b,c = 1,2.5,True

Here, two integer objects with values 1 and 2 are assigned to variables a and b respectively, and one string object with the value "john" is assigned to the variable c.

The values will be assigned in the order in which variables appears.

**2.2 Tokens and their types :**

* Tokens can be defined as a punctuator mark, reserved words and each individual word in a statement.
* **Token is the smallest unit inside the given program.**

There are following tokens in Python:

**identifiers operator literals Keywords constant**

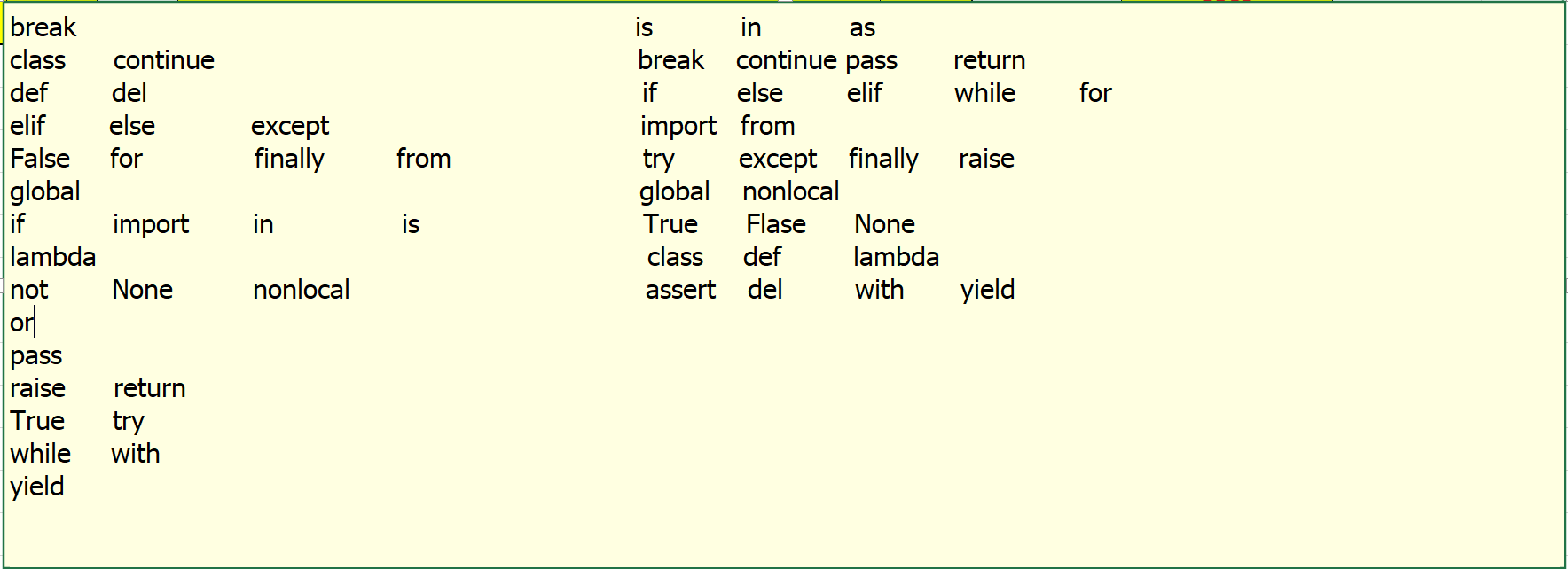
**--------------------------------------------Identifiers Operator Literals**

**x = 10**

**Keywords**

**a.Keywords** :

Are **reserved words** and has specific meaning in a language and they cannot be used as ordinary identifiers.



**b.Identifiers** :

An **identifier is a variable name.**A Python identifier is a name used to identify a variable,

function name,

class name,

module name or

other object name.

An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9)

### Rules for writing Identifiers

There are some rules for writing Identifiers.

But first you must know **Python is case sensitive.** That means **Name** and **name** are two different identifiers in Python.

Here are some rules for writing Identifiers in python.**PEP8 standards**

<https://www.python.org/dev/peps/pep-0008/>

\_age emp\_nameemp\_salstudent\_id

* Identifiers can be combination of uppercase and lowercase letters, digits or an underscore(\_). So **myVariable**, **variable\_1**, **variable\_for\_print**all are valid python identifiers.
* An Identifier can not start with digit. So while **variable1** is valid, **1variable** is not valid.
* We can’t use special symbols like !,#,@,%,$ etc in our Identifier.
* Identifier can be of any length.

msg = ‘My name is Madhu’

Though these are hard rules for writing identifiers, also there are some naming conventions which are not mandatory but rather good practices to follow.

**Naming Conventions:**

1. **Class names** start with an **uppercase** letter.
2. All **other** identifiers start with a **lowercase** letter.**\_age \_\_age\_\_**
3. Starting an identifier with a single leading underscore indicates the identifier is private.
4. If the identifier starts and ends with two underscores, than means the identifier is language-defined special name.
5. While **c = 10** is valid, writing **count = 10** would make more sense and it would be easier to figure out what it does even when you look at your code after a long time.
6. Multiple words can be separated using an underscore, for example this\_is\_a\_variable.

**c = 10**  **emp\_id = 10**

**x = 123.45 mobile\_bill = 123.45**

**c.Literals** :

In computer science, a literal is a notation for **representing a fixed value**in source code.

name = “Madhu” --- String literal

age = 10 ---integer literal

sal = 123.45 – float literal

age = 10+20

The **literals** include the string, unicode string, integer, float, list, tuple and dictionary types

**d.Operators** :

Operators are special symbols in **Python** that carry out arithmetic or logical computation.The value that the **operator** operates on is called the operand

**>>>2+3**

**5**

Here **+** is an **operator** which is performing arithmetic computation.

**2** and **3** are the **operands** and

**5** is the output of the operation.

## Constants

A constant is a **type of variable** whose value cannot be changed. It is helpful to think of constants as containers that hold information which cannot be changed later.

Non technically, you can think of constant as a bag to store some books and those books cannot be replaced once placed inside the bag.

### Assigning value to a constant in Python:

age = 10 # **variable**

PI = 3.14 22/7 # PI **constant**

*GRAVITY = 9.8* # GRAVITY is a **constant**

*WEEKS* = {MON,TUE,WED,THU,FRI,SAT,SUN}

Work of Stack Memory

The allocation happens on contiguous blocks of memory. We call it stack memory allocation because the allocation happens in the function call stack. The size of memory to be allocated is known to the compiler and whenever a function is called, its variables get memory allocated on the stack.

It is the memory that is only needed inside a particular function or method call. When a function is called, it is added onto the program’s call stack. Any local memory assignments such as variable initializations inside the particular functions are stored temporarily on the function call stack, where it is deleted once the function returns, and the call stack moves on to the next task. This allocation onto a contiguous block of memory is handled by the compiler using predefined routines, and developers do not need to worry about it

def func():

# All these variables get memory

# allocated on stack

a = 20

b = []

c = ""

Work of Heap Memory

....The memory is allocated during execution of instructions written by programmers.

....Note that the name heap has nothing to do with heap data structure.

....It is called heap because it is a pile of memory space available to programmers to allocated and de-allocate.

....The variables are needed outside of method or function calls or are shared within multiple functions globally are stored in Heap memory.

# This memory for 10 integers

# is allocated on heap.

a = [0]\*10